Remarks

This REQUEST FOR CONTINUED EXAMINATION and REPLY is in response to the Office Action mailed October 1, 2007. A Petition for Extension of Time is submitted herewith, together with the appropriate fee. No fee is due for the addition of new claims.

I. Summary of Examiner's Rejections

In the Office Action mailed October 1, 2007, Claims 1-4, 6-11, 13-18, 20-21 were rejected under 35 U.S.C. 103(a) as being unpatentable over McNeely et al. (U.S. Patent No. 7,117,411, hereafter McNeely) in view of Dubovsky (U.S. Publication No. 20030055836). The Examiner also indicated various informalities in Claims 8, 15 and 18.

II. Summary of Applicant's Amendment

The present REPLY amends Claims 1, 8, 15 and 18; and adds new Claims 22-24, leaving for the Examiner's present consideration Claims 1-4, 6-11, 13-18 and 20-24. Reconsideration of the Application, as amended, is respectfully requested.

III. Objections to the Claims

In the Office Action mailed October 1, 2007, the Examiner noted various informalities in Claims 8, 15 and 18. Applicant thanks the Examiner for pointing out the informalities, and believes that the claims as presented above include the correct text. Applicant regrets any inconvenience this may have caused.

IV. Claim Rejections under 35 U.S.C. §103(a)

In the Office Action mailed October 1, 2007, Claims 1-4, 6-11, 13-18, 20-21 were rejected under 35 U.S.C. 103(a) as being unpatentable over McNeely et al. (U.S. Patent No. 7,117,411, hereafter McNeely) in view of Dubovsky (U.S. Publication No. 20030055836).

Claim 1

Claim 1 has been amended by the current Response to more clearly define the embodiment therein. As amended, Claim 1 defines:

1. (Currently Amended) A system that provides a generic user interface testing framework, and allows a user to test and debug graphical user interfaces for software applications under development, comprising:

a computer including a computer readable medium, and a processor operating thereon;

a software application source code, stored on the computer readable medium, wherein the software application source code defines a software application under development, including a graphical user interface as part of the software application, and wherein the software application source code executes on the computer to display its graphical user interface;

one or more different software test tools that can be invoked by a user to perform testing operations on the graphical user interface that is displayed while the software application is running, wherein each of the one or more different software test tools understand their own tool-specific scripting language;

a test case input file stored on the computer readable medium, that contains a plurality of generic interface commands that are abstractions independent of any tool-specific scripting language, wherein the test case input file can be edited and reused as necessary by the user to specify different generic interface commands for testing against a software application's graphical user interface in any of the different software test tools; and

an interpretive engine that executes on the computer, and that includes a plurality of dynamically loaded libraries corresponding to the plurality of different software test tools, and including a library for each of the one or more different software test tools, wherein the interpretive engine receives the generic interface commands defined in the test case input file, determines which software test tool the user is currently using, loads required libraries to map the generic interface commands to corresponding tool-specific testing operations, uses the software test tool to perform the testing operations on the software application's graphical user interface including translating the generic interface commands to tool-specific commands, and reports to the user the success or failure of the testing operations.

Claim 1 has been amended to more clearly define the embodiment therein as comprising one or more different software test tools that can be invoked by a user to perform testing operations on a graphical user interface, wherein each of the one or more different software test tools understand their own tool-specific scripting language. A test case input file contains a plurality of generic interface commands that are abstractions independent of any tool-specific scripting language. The test case input file can be edited and reused as necessary by the user to specify different generic interface commands for testing against a software application's graphical user interface in any of the different software test tools. An interpretive engine that includes a library for each of the one or more different software test tools, receives the generic interface commands, determines which software test tool the user is currently using, and translates the generic interface commands to tool-specific commands.

The advantages of the embodiment defined by Claim 1 include that the system maps generic interface commands into tool-specific testing operations. While automated test development systems, suites and tools have been developed, the typical approach of such automated test development tools requires that the operator have knowledge not just of the test-tool-specific scripting language and environment, but also the specific features and idioms of the vendor-specific tool environment. The learning curve for these testing tools can be significant, and is often of use solely with that one tool. In accordance with the embodiment of Claim 1, the system insulates testers from the tool-specific testing operations of a particular testing tool, so that effort spent testing the build of a software application can be reduced.

McNeely discloses an apparatus and method for a communications network test system. As disclosed therein, the test system includes device-specific communication interface packages that map generic commands to device-specific commands. (Abstract). Prior to testing a network device, a test case is selected for execution by a test operator. (Column 15, lines 25-30). The information contained within the test case file identifies one or more of the devices under test. (Column 15, lines 35-36). Consequently, one of the primary objectives or goals of the testing process prior to deployment is to identify and resolve any potential problems in the provisioning of network services that may result from the diverse nature of the collection of network elements. (Column 1, lines 33-37). As further disclosed therein, for the purposes of illustration, the detailed discussions presented herein are limited to communication with those devices that employ command line interfaces, since this type of interface is currently the most widely deployed interface type. Communication with GUI-based devices can occur via a graphical user interface if a suitable GUI tester is added via a new package. (Column 1, lines 33-37).

Dubovsky discloses methods for generating data structures for use with an environment based data driven test engine for computer programs which have a graphical user interface (GUI). The methods are used in connection with a scriptable GUI test tool. The tool generates a GUI map (or includes a utility which generates a GUI map), at least one environment definition (parameter) file, at least one test data (driver) file, and an automated test engine. A separate environment definition file is provided for each feature of the GUI. Each environment definition file provides the abstract details required by the test engine in order to support common processes for different applications. (Abstract). As further described therein, it is another object of the invention to provide the maximum flexibility for test case generation, maintenance and execution required during the development and test cycle of a GUI software project. (Paragraph [0015]).

It appears from the above description that, in McNeely, a test case is used to test network devices. A test case selected contains information identifying one or more devices. Any device not contained within the test case must be added to the current test case, or a different test case has to be chosen that contains that device. Consequently, it is the test operator who directs the system to load the appropriate libraries that are associated with the particular devices under test since the system will not load these libraries unless the test operator has included these devices in the test case. McNeely also appears to disclose that while some devices are command-line based, other devices may be GUI-based, and that these GUI-based devices can be similarly tested if a suitable GUI tester is added via a new package.

It also appears from the above description that, in Dubovsky, the system appears to allow for the development of software test scripts that can be reused with different software projects. This can include components that may be specific to a particular software project, and also components (such as GUI components) that may be common to several different software projects.

However, Applicant respectfully submits that, in both McNeely and Dubovsky, it appears the systems disclosed therein are directed toward applying similar tests to different test subjects, i.e. to different network devices, different software projects, or different components of a software project. To accomplish this task, both McNeely and Dubovsky appear to disclose scripts that can be reused for each new test subject, to provide an overall means of using a testing script between a somewhat fixed testing environment and different test subjects.

In contrast, the embodiment of Claim 1 provides a means of using a testing script between different testing environments. As described above, while automated test development systems, suites and tools have been developed, the typical approach of such automated test development tools requires that the operator have knowledge not just of the test-tool-specific scripting language and environment, but also the specific features and idioms of the vendor-specific tool environment. The test-tool-specific code produced by action recorders is often relevant only to the specific parameters and environment(s) where the recorder was employed.

To more clearly define the embodiment therein, Claim 1 has been amended to define the embodiment as comprising one or more different software test tools that can be invoked by a user to perform testing operations on a graphical user interface, *wherein each of the one or more different software test tools understand their own tool-specific scripting language*. A test case input file contains a plurality of *generic interface commands that are abstractions independent of any tool-specific scripting language*. The test case input file can be edited and reused as necessary by the user to specify different generic interface commands for testing against a software application's graphical user interface in *any of the different software test tools*. An interpretive engine that includes a library for each of the one or more different software test tools, receives the generic interface commands, determines which software test tool the user is currently using, and *translates the generic interface commands to tool-specific commands*.

An advantage of the embodiment defined by Claim 1 includes that, if a new testing tool is used to replace an existing testing tool, then all of the directives of test logic can be retained, and only the interpretive engine and the function libraries need be re-implemented in the new scripting language. For example, switching from one testing tool to another requires only changes to the interpretive engine and/or the function libraries; the directives themselves, which can be saved as test files, spreadsheets, or some other format, may be re-used.

In view of the comments provided above, Applicant respectfully submits that the embodiment defined by Claim 1 is neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

Claims 8 and 15

The comments provided above with respect to Claim 1 are hereby incorporated by reference. Claims 8 and 15 have been similarly amended to Claim 1 to more clearly define the embodiment therein. For similar reasons as provided above with respect to Claim 1, Applicant respectfully submits that Claims 8 and 15, as amended, are likewise neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

Claims 2-4, 6-7, 9-11, 13-14, 16-18 and 20-21

Claims 2-4, 6-7, 9-11, 13-14, 16-18 and 20-21 depend from and include all of the features of either Claim 1, 8 or 15 respectively. Claims 2-4, 6-7, 9-11, 13-14, 16-18 and 20-21 are not addressed separately but it is respectfully submitted that these claims are allowable as depending from an allowable independent claim, and further in view of the amendments to the independent claims and the comments provided above. Applicant respectfully submits that Claims 2-4, 6-7, 9-11, 13-14, 16-18 and 20-21 are similarly neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

V. Additional Amendments

Claims 22-24 have been newly added by the present Reply. Applicant respectfully requests that new Claims 22-24 be included. in the Application and considered therewith.

VI. Conclusion

In view of the above amendments and remarks, it is respectfully submitted that all of the claims now pending in the subject patent application should be allowable, and reconsideration thereof is respectfully requested. The Examiner is respectfully requested to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

Enclosed is a PETITION FOR EXTENSION OF TIME UNDER 37 C.F.R. §1.136 for extending the time to respond up to and including April 1, 2008.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 06-1325 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: April 1, 2008 By: / Karl Kenna/ .

Karl Kenna
Reg. No. 45,445

Customer No.: 23910
FLIESLER MEYER LLP
650 California Street, 14th Floor
San Francisco, California 94108
Telephone: (415) 362-3800